

LA-UR--85-854

DE85 009579

TITLE: The Los Alamos Automated Configuration Accounting System: A Proposal

AUTHOR(S): G. Cort and J. A. Goldstone

SUBMITTED TO: Softool User's Group Meeting
c/o Mr. G. Kim Jenkins
LTV Aerospace and Defense Company
P.O. Box 225907
Dallas, Texas 75265

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of the published form of this contribution, or to allow others to do so, for U.S. Government purposes. to publish or reproduce

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

 Los Alamos National Laboratory
Los Alamos, New Mexico 87545

The Los Alamos Automated Configuration Accounting System: A Proposal*

G. Cort and J. A. Goldstone
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Introduction

An effective configuration accounting (CA) effort is vital to the success of any software configuration management plan. The CA effort maintains information that details the current status of the software project as well as information that checkpoints the project's history. This information can be used to great advantage by project managers to plan and schedule project activities, to evaluate management strategies, and to serve as a knowledge base for future projects.

Unfortunately, this process is often tedious, time-consuming, and extremely labor-intensive. As essentially every significant event of a project must be documented, recorded and cataloged, this effort can produce large volumes of paper, which themselves require some form of management. For projects with relatively small resources, these overheads can become quite daunting, often resulting in abandonment of a formal configuration accounting effort.

In this paper, we propose an automated system for managing a modest configuration accounting effort associated with the development and maintenance of a large real-time data acquisition system. For reasons presented in the following sections, we propose building this system around the Softool Change and Configuration Control (CCC) environment.

Background and Scope of the Problem

The Data Acquisition Command Language (DACL) currently under development at the Los Alamos Weapons Neutron Research/Proton Storage Ring Facility is a large real-time data acquisition software system supporting neutron scattering experiments. Although the development team is relatively small (4-5 people), a software configuration management plan was approved for the project, and a configuration accounting effort was specified as a primary component of that plan. Because of the small development staff, and owing to the absence of independent auditing and configuration accounting entities, one team member assumes the role of configuration manager in addition to the usual development responsibilities.

The DACL configuration accounting specification defines five separate forms, known as Software Action Requests (SAR's), which are used to document the life cycle of each software configuration item in the DACL system. Several of the SAR's provide a mechanism for reporting, tracking, and disposing of software change requests and discrepancy reports. Others are employed to document the configuration auditing effort, and the last records transactions between team members and the various libraries of software configuration items. The DACL CA effort is responsible for the identification,

MASTER

EMC

cataloging, tracking, and disposal of these forms for every configuration item over the entire software life cycle.

The PACL configuration management plan prescribes that each class of SAR be maintained in a series of catalogs. In addition to a chronological log (ordered by date of receipt), the plan requires separate logs for pending (unassigned), open (in-progress) and closed (completed) requests, and a catalog of requests organized by configuration item. As the development/maintenance effort progresses, requests migrate between the various logs in their class, ultimately finding their way into the "closed requests" log.

The problems associated with this system, from the perspective of the small project, stem from the requirement that all modifications and updates to these requests must be manually performed. Filing the forms, maintaining the logs and generating reports on their contents also require the personal attention of a human being for even the most routine operations. The associated overheads can become so severe as to force abandonment of some aspects of the CA effort. This, of course, reduces the effectiveness of the remainder of the CA program and can actually result in making some of the remaining CA tasks more difficult to perform.

The Los Alamos Strategy

Our proposal for automated SAR management utilizes CCC to perform many routine operations completely without human intervention. In addition, the system takes advantage of the simple data base features of the CCC environment to organize and store the various classes of SAR's, as well as to facilitate the generation of a wide variety of CA reports.

Figure 1 shows a typical SAR: the Software Change Request. A template for this form is electronically filed on the support computer system and can be accessed by any system user. The template can be completed using a word processor and transmitted to the configuration manager by electronic mail. As part of the transmission process, existing software generates a unique identifier for the SAR. This identifier is forwarded to the configuration manager for incorporation into the ID field (upper right) of the SAR. Also upon receipt, the configuration manager is responsible for updating the first section of the ACTION block (lower right) to record the date of receipt. At the times when the SAR is assigned and reviewed, additional entries must be made. Also, after each such transition, the appropriate logs must be updated with fresh copies of the modified SAR.

We propose that the vast majority of the modifications enumerated above can be performed automatically by CCC macros. Beginning with the transmission process, a CCC macro could be invoked (probably through the CCCBATCH facility) to write the SAR identifier and the receipt date directly into the appropriate fields of the inbound form. The macro could also perform the task of cataloging the SAR into the appropriate data structure in a CCC data base and automatically print a copy of the newly arrived request. Other macros could be invoked to supervise the transition of a SAR between ACTION states.

CHANGE REQUEST			
Title:		ID:	
		Date:	
Originator:	Group:	MS:	Phone:
<input type="checkbox"/> System Software		<input type="checkbox"/> Application Software	
<input type="checkbox"/> Hardware		<input type="checkbox"/> Documentation	
Configuration Item:	System Version:	Priority: <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3	
Description of Change:			
Justification for Change:			
DO NOT WRITE BELOW THIS LINE			
Remarks/Analysis			
		ACTION	BY
		DATE	
		RECEIVED	
		TO	
		ANALYST	
		TO	
		PROGRAMMER	
<input type="checkbox"/> Approved Priority: <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3		ACCEPTED	
<input type="checkbox"/> Approved with modification Priority: <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3		TO	
<input type="checkbox"/> Rejected		ORIGINATOR	
Analyst:		Time expanded:	
Programmer:		Time expanded:	

Figure 1. A typical SAR template.

Although a system such as this should prove helpful in reducing the overheads associated with SAR cataloging and modification, its major strength lies in the area of report generation. The fundamental problem associated with report generation in the CA environment is that the information contained within a class of SAR's cannot be readily accessed in an automated fashion. Hence, reports can only be compiled by the laborious task of manually examining existing SAR's and extracting those that contain relevant information. Summary reports are even more difficult to generate as entirely new documents must be manually created.

The CCC-based system can eliminate these difficulties for most common reports by maintaining (as CCC text characteristics) much of the key information contained within a SAR. These characteristics can be used to specify reports to be automatically generated by the CCC system. Most significantly, these characteristics can be automatically maintained by the CCC macros that perform the SAR cataloging and updating tasks.

For the DACL system, we require the capability to generate reports based upon the following properties of each class of SAR's: identifier, status (pending, open, closed), affected configuration item, and date of last action. A brief descriptive abstract should also be available for each SAR. It is also necessary to be able to recreate any SAR at any point in its history.

All of the information specified above is easily accommodated within the characteristics of a CCC text. The SAR identifier is stored as the text name. The configuration item can be read from the SAR by a macro and stored as the text type. The CCC macros that process a SAR upon receipt or change of status can automatically assign a change status value (pending, open, closed) to the change name text attribute. The date of last action is automatically recorded by CCC. The descriptive abstract can be obtained by (automatically) extracting the title field from the SAR and assigning it to the change description characteristic. If archiving is enabled, prior versions of each SAR will be retained and can be reviewed on demand.

With the vital statistics of each SAR stored as described above, the LISTSTRUCTURE and LISTCHANGE commands become extremely powerful CA report generation tools. By exploiting the simple data base features of the CCC environment, these commands allow a great variety of reports to be generated without the development of additional specialized software. Under such a system standard reports can be routinely generated with minimal human intervention. Indeed, the idea of using a permanently resident batch job to generate a suite of reports at fixed time intervals is very appealing.

Exploiting CCC Version 2.0 Features

Although the system proposed above could, in principle, have been developed under previous releases of the CCC environment, it is our contention that CCC Version 2.0 is the first implementation under which an automated project of this scope and complexity can be seriously contemplated. In this section we discuss some of the Version 2.0 features that qualify CCC as an effective development language for this, or any other, large application.

From the perspective of automated applications, the introduction of an error-handling facility is probably the most important new feature. The previously implemented policy of forcing macro termination upon encountering an unforeseen or simply unusual condition results in applications that are difficult to develop and extremely fragile. In contrast, by allowing the developer to handle selected conditions explicitly, applications developed under the Version 2.0 environment should be simpler and far more robust.

The addition of conditional branching and looping to the macro facility should also significantly enhance the programmer's ability to develop large applications. Conditional branching was previously unavailable (and sorely missed). The new looping constructs augment the existing, but limited facility. The implementation of each of these features as structured constructs should promote the unambiguous mapping of the application problem space into a programmed solution, resulting in programs that are easier to read and to understand.

The addition of symbols for storing information is expected to provide a double benefit. First, the use of symbols should eliminate the need for inserting and subsequently deleting text data structures to store intermediate results or control information. Second, by largely eliminating the use of texts for storing "temporary" information, we expect that the accompanying decrease in file accesses will result in a significant enhancement of performance.

Other Version 2.0 features will doubtless also prove extremely useful. The ability to perform prompting from within a macro will introduce a great amount of flexibility into the design of human interfaces. Enhanced report generation will permit truly customized reports to be compiled. The library of predefined macros will permit complex operations to be performed at a high level and will also allow access to low level data structures. In all, we expect CCC to become a very comfortable application development environment.

Conclusion

We have presented the functional capabilities required of an automated system for administering the configuration accounting effort of a software development project. Although this proposal is advanced from our perspective as a small-team project, the lack of implementation details allows this approach to be considered by projects of any size. We hope that we have imparted an awareness of the tremendous gains realized from automating CA (and all other configuration management) activities and have demonstrated the powerful tools for obtaining this goal which can be built from the Version 2 CCC environment.

The specifics contained within our proposal presume a small-team software development environment. In our view, the benefits of configuration accounting (and configuration management in general) are accessible to small projects only through automation of these activities. Regardless of the size of the project, automating configuration management activities frees project participants from tedious and mundane tasks, thus increasing im-

mediate and long-term productivity. Incorporation of tools which dramatically improve the software management process is essential for all size projects.

We conclude by promising to report upon and evaluate the detailed implementation of this proposal at a subsequent Softool User's Group Meeting.